Res. Lett. Inf. Math. Sci., 2004, Vol.**6**, pp83-96
Available online at http://iims.massey.ac.nz/research/letters/

Genetic Programming: The ratio of Crossover to Mutation as a function of time

David R. Munroe

Institute of Information & Mathematical Sciences
Massey University at Albany, Auckland, New Zealand.

This article studies the sub-tree operators: mutation and crossover, within the context of Genetic Programming. Two standard problems, symbolic linear regression and a non-linear tree, were presented to the algorithm at each stage. The behaviour of the operators in regard to fitness is first established, followed by an analysis of the most optimal ratio between crossover and mutation. Subsequently, three algorithms are presented as candidates to dynamically learn the most optimal level of this ratio. The results of each algorithm are then compared to each other and the traditional constant ratio.

1 Introduction

The concepts of Genetic Programming (GP) are derived from the biological world and Darwin's theory of evolution [1]; in particular, adaptation to environmental parameters via survival of the fittest - the move from a general population to a specialised population. Genetic Programming seeks to emulate the adaptation process by generating a series of programs and selecting the better ones to create the next generation of programs via some method of reproduction. Obviously, two things are required by the algorithm for GP to work. Firstly, a sensible method of evaluating the programs needs to be in place. Secondly, a means of reproducing programs needs to be able to be used.

The methods of program reproduction generally used in GP are taken loosely from biology: namely crossover and mutation. Cloning, or an exact copy, is another operator that is often included. While these are the more 'useful' reproducers, other reproduction operators are also possible [3].

In the literature it is customary to have a fixed ratio of crossover to mutation throughout the entire Genetic Programming simulation. The accepted 'best' ratio

is held to be $\sim$ 95% Crossover: 5% Mutation. As shown in [4], however, the relationship between the best ratio, population size, and problem specification is not as clear cut as required to justify an arbitrary ratio between the operators.

Interestingly, mutation rates do not remain constant over time within a living populations' development. Rather, mutation rates can evolve in asexual populations when beneficial mutations occur [5]. The adaptive significance may be limited long term, but the short term effects may be highly influential when examined over that shorter period. Changes in mutation behaviour, even short term deviances, may be the key to determining how local maxima (or minima) in population development can be broken.

One possible way to speed up the adaptation in GP, reduce the impact of local maxima, and thus reduce the overall evolution time, is to vary the ratio of crossover to mutation. The justification for this is based on the differences between the two operators. Obviously, in biology the differences between sexual and asexual reproduction can be distinguished. In GP the distinction is not as clear as the crossover points are random and it can be argued that, in both cases, random code is being swapped with random code. However, one importance fundamental difference is: crossover only can use what material is already contained in the population - mutation can generate code using any function including those that may have been previously (perhaps erroneously) selected against.

Provided that the Genetic Programming environment is not too simplified, the dominance of sexual reproduction in the biological world could be expected to follow through to the dominance of the 'crossover' operator in the Genetic Programming paradigm. Thus, we could expect that the optimal level of the operators would tend toward a proportionally high level of crossover and a relatively low level of mutation. This is consistent with the standard rate of 90+% Crossover. John Koza in his work [3] uses only a very low proportion of mutation and suggested a maximum level of 10%. Other researchers have reported similar findings on the value of mutation as a reproduction operator.

1.1 Caveats

Any improvement in the function of the operators, and thus justifying increasing the ratio in the direction of improvement, is founded on these operators being applied randomly to programs. This random application of reproductive methods does not occur in biology. Organisms are generally designed to respond best to either asexual or sexual reproduction. The end result of this is that there may be a finite level of improvement to altering operator ratios based on random selection and application.

The implementation of the operators in this analysis, which allows the application of the operator at any point in the generated programs with equal probability, may not be the best way to design either mutation or crossover. This operator design is simple and is in accordance with the choice of program storage as a prefix function list, as suggested in [2], rather than a more 'costly' program tree. This

change in implementation, however, has reduced the destructivity of the mutation operator as the tree mutation operator selected mutation points within the tree 90% of the time.

2 Methodolgy

Two standard problem domains are considered for the GP algorithm to solve. The first is a linear symbolic regression problem: Figure 1:

$$y = 2 \times \pi \times x_1 + 0.814 \times x_2^3 + error \tag{1}$$

where error $\sim N(x_1, 2)$.

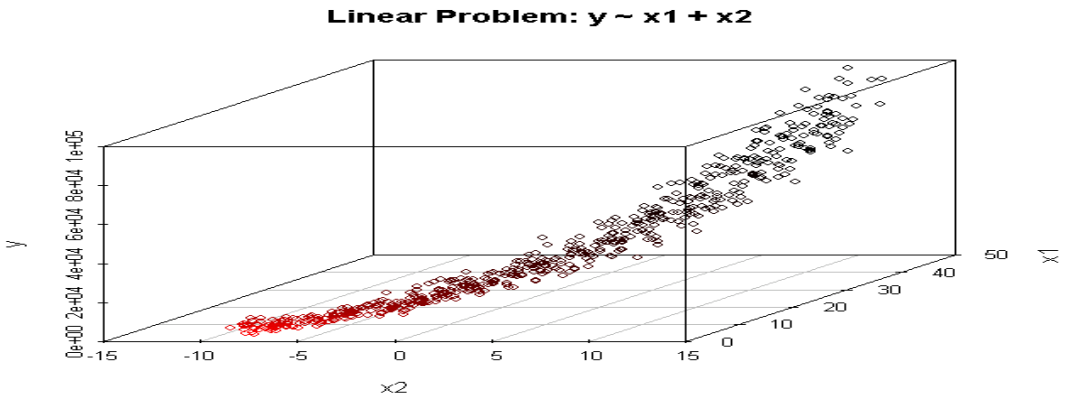

Figure 1: Linear Problem Graph

The second problem is a nonlinear problem where there are two subgroups within the data. Again, the solution can be expressed as a function of x_1 and x_2: Figure 2.

$$\begin{aligned} y = (x_1 - x_2) \times (x_1 - x_2) + error; x_2 \geq x_1, error \sim N(x_2, x_2/8.0) \\ y = 200 + 2 \times x_1 + error; x_1 < x_2, error \sim N(x_1, x_1/4.0) \end{aligned} \tag{2}$$

where $x_1 \sim exp(1/4) + 4.0$ and $x_2 \sim N(8, 4)$.

Although other methodologies produce 'better' solutions, in a shorter time-frame, to these problems, the principles derived from these two domains can be extended into any problem space.

2.1 Influential factors

A number of factors were hypothesised to influence operator development. Some of these items listed are also considered in [4], but additional categories have been added to this original list to account for use of a parallel system. The more influential

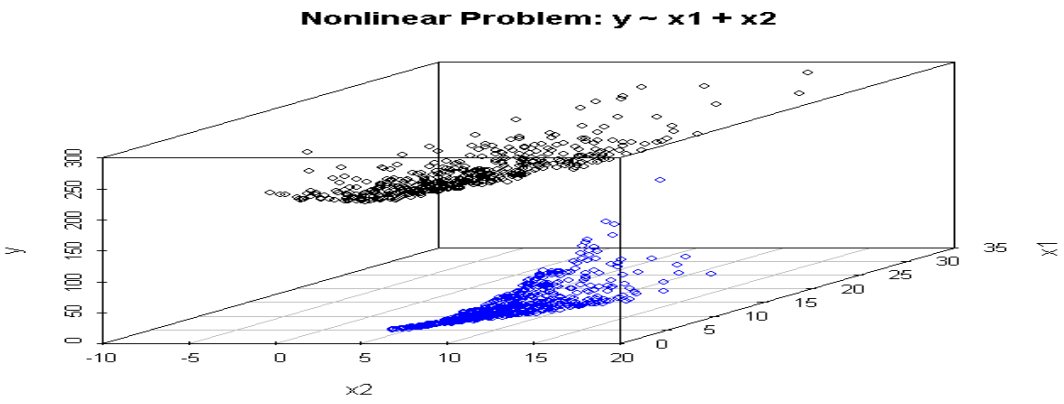

Figure 2: Non-Linear Problem Graph

items are listed as follows: Problem domain, Available Functions, Maximum Program Size, Population Size, Number of Populations, Initial Operator Proportions, Program Selection Method, and the Program Migration Rate.

In order to reduce the size of the experiment, both the number of factors considered and the range of levels these factors could take had to be kept to a minimum. Thus the factors considered to varying extents were: Population Size, Number of Populations, Initial Operator Propotions, and the Program Selection Method.

3 Operator analysis

Before varying the crosover to mutation ratio, it is important to confirm that the operators are indeed practically different - or any variation would be nonsensical. Thus the fitness distribution of each operator is examined followed by analysis of the operator behaviour over time. The focus in this latter half is to find the 'optimal' ratio of the operators and whether changing the operator ratios is justifiable.

3.1 Distribution analysis

For a series of generations, the operators can be applied and the resulting fitness recorded for each program. A statistical test for the difference of distributions and means could then be applied for each generation.

Hypothesis being tested: the distribution of the fitness for programs created by crossover at generation, G, is the same as the fitness distribution for programs generated by mutation at G for all G. The alternative hypothesis being that the distributions are not the same for at least one G.

3.1.1 Linear problem

The Kolmogorov-Smirnov (K-S) test on the two samples for the thirteen nodes is presented in Figure 3. The K-S P-Values have been plotted against time for each node - the dashed lines are for the first run, the solid lines for the second. For comparison, the plot of the analysis of the combined runs has also been included.

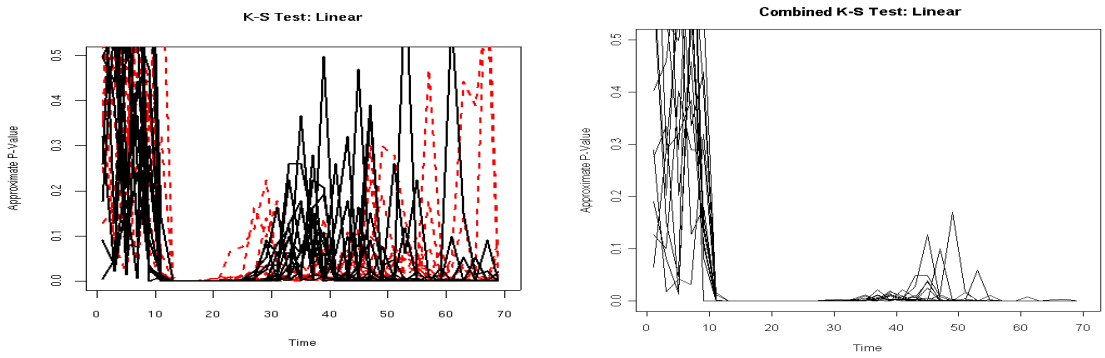

Figure 3: K-S Test: Linear Regression

Although the second plot is clearer to read, it is clear that combining the runs has increased incorrectly amplified the difference between the two operators - especially for the later generations. The difficulty of the problem - in terms of Genetic Programming - is clearly evident in the first plot: LHS Figure 3. If swapping random code gives as good a result as swapping code that is 'more useful' then it seems that the program is really not learning anything.

The test shows that the two operators are very similar initially for most nodes. Indeed, this similarity does not really cease till around generations $10-12$ for either run. The distribution fitness of the operators seems to be only consistently different across all nodes for around 10 generations. After this point, the measurement of program fitness fails to adequately distinguish between the programs produced by crossover and those produced by mutation.

3.1.2 Nonlinear problem

The Kolmogorov-Smirnov (K-S) test on the two samples for the thirteen nodes is presented in Figure 4. The graph has been edited slightly from the original R plot to make it clearer. In particular, critical regions have been attempted to be shown rather than a line plot.

The areas are generally selected so that the most dominant sample is shown. However, during the generations 10-35 both samples were extremely volatile - even though only the first sample is shown.

The test shows that the operators perform similarly for some generations but not for all. According to the K-S test, the differences between the operators do not

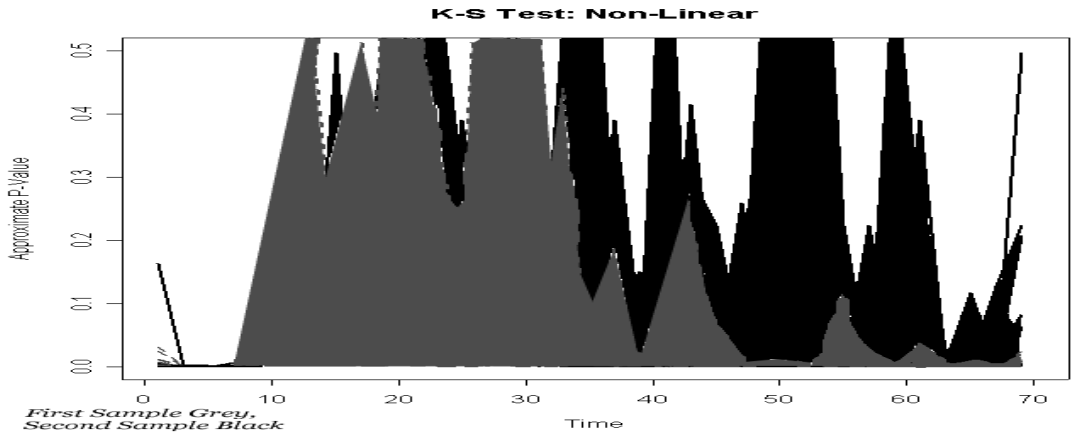

Figure 4: K-S Test: Non-Linear Problem

necessarily grow with time. Instead, the results in Figure 4 suggest that there are three stages in the GP algorithm. In the initial stage the operators are very different - mutation outperforms crossover. The second block shows the operators as being essentially the same, while the final stage indicates that the operators are different again - crossover outperforms mutation. Note that this final stage is only evident in the first sample.

More likely, however, these stages could be viewed as periods of learning, where crossover is dominant, or periods of blind search, where neither operator is dominant. The dominance of mutation also implies a blind search.

3.2 Time analysis

To test for a constant operator ratio, record the current 'best' ratio of crossover to mutation and use it for the next generation of programs. A statistical test for $Ratio_t = Ratio_{t+1} = \ldots = Ratio_{t+n}$ could then be examined. The test requires an unbiased estimator of the 'best' proportion. The estimator chosen was the ratio between the operators from the programs in the upper quartile.

Hypothesis being tested: $Ratio_t = Ratio_{t+1} = \ldots = Ratio_{t+n}$. Alternative Hypothesis: $Ratio_t \neq Ratio_{t+h}$ for some $0 < t + h < n$.

Unfortunately, as the samples are not independent, this hypothesis is not easily testable. The runs are independent of each other, but the observations within each run are not independent. Thus, the actual degrees of freedom available are greatly overestimated if a test is naively applied without taking the dependence into account.

One to determine whether a series is a function of time is to see whether a series is stationary or not. Essentially, the model:

$$Ratio_t = f(t) + e_t \tag{3}$$

where $f(t)$ is some function of time. The easiest way to prove non-stationarity is to

show that the series has a unit root.

3.3 Linear problem

The Phillips-Perron unit root test was used as "The Phillips-Perron method estimates the non-augmented Dickey-Fuller test equation, and modifies the t-ratio of the α coefficient so that serial correlation does not affect the asymptotic distribution of the test statistic." [1].

Table 1: 13 Nodes 14K Max: Phillips-Perron Unit Root for Node 2 Test

Run	DF Test Value	P-Value	Run	DF Test Value	P-Value
1	-0.4476	0.9818	16	-0.2368	0.99
2	-0.4371	0.9823	17	-0.3202	0.9875
3	-0.4542	0.9815	18	-0.4397	0.9822
4	-1.0552	0.9237	19	-0.8249	0.9554
5	-0.3697	0.9853	20	-0.6173	0.9737
6	-1.5145	0.7742	21	-0.1941	0.99
7	-0.0435	0.99	22	-0.92	0.9428
8	-0.1732	0.99	23	-0.4468	0.9819
9	-0.0409	0.99	24	-0.5027	0.9794
10	-0.7959	0.958	25	-0.6259	0.973
11	-1.3697	0.8334	26	-2.6343	0.3163
12	-1.7879	0.6616	27	-0.3727	0.9851
13	-0.0239	0.99	28	-0.2572	0.99
14	-2.0589	0.5516	29	-1.1723	0.9054
15	-1.861	0.6325	30	-1.1759	0.9048

The results of the Phillips-Perron test for Node 2, shown in Table 1, confirm that the overall average is a function of time. The model does have a unit root (H_0 = Unit Root) for all runs. Thus, the ability to predict the next optimal value is not necessarily possible. The best prediction could just be what we currently observe now - which is hardly a useful prediction in terms of a trend.

The results for other nodes all showed that there was a unit root for each of the runs. The optimal series is not stationary and, therefore, is not constant in time.

3.4 Non-linear problem

As with the linear problem, the conclusion of all the tests conducted was that there is some dependence of the $Ratio_t$ in time. A further result that became self-evident

[1] Eviews 4.1 Help: Unit Root Tests

was that a blanket generalisation cannot be made for the nodes or runs. Each run is different depending on the known factors and error governing that run. Basing a model on general behaviour may produce a better result than the assumption of a constant ratio, but a better approach may be to let the ratio self-adjust to the unique circumstances per run.

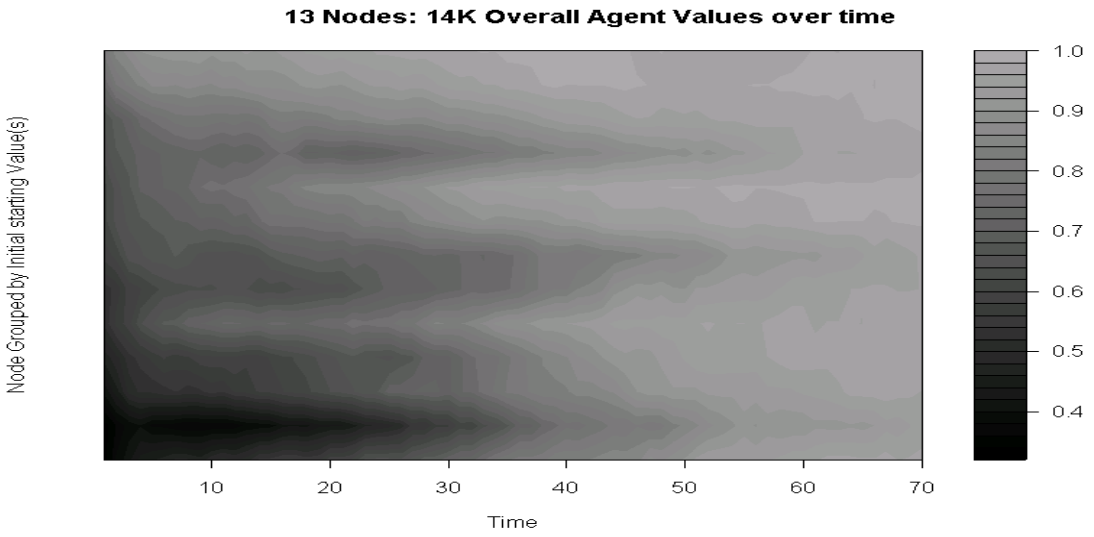

Figure 5: Overall Operator Behaviour by Node

If the contour plot of the overall operator ratio over time is examined, Figure 5, there is a definite general trend evident with all series. The contour plot is shown for the 13 node case with the first node having a population size of 14K. The other tests with a different number of nodes and/or different population sizes show similar results.

4 Operator evolution

The optimal levels of mutation to crossover vary depending on a number of factors. An additional factor not considered relevant for a changing ratio, but necessarily considered for a static ratio, is the Number of Generations [4]. In addition, the relationship between the population and the optimal operator levels is vague, and it is obvious that a static ratio cannot account for such variation. Therefore, an algorithm which optimised the operator ratio would be desirable as no prior knowledge regarding the best operator ratio for the current combination of factors would be required.

The results from the previous section showed that the overall ratio approaches the optimal level of 95%+ crossover as time goes on. However, the rate of convergence does change with time.

4.1 Models for changing ratios

4.1.1 Random walk model

The Random Walk model, RW, for changing the operator ratios assumes no real learning is possible (4).

$$Ratio_{t+1} = Ratio_t \tag{4}$$

The best ratio for the next time point is assumed to be whatever is observed in the upper quartile at the current time point. The results for this 'learning' algorithm provides the baseline for any other learning algorithm considered. Note that this model is only a true random walk if there is no relationship between crossover and mutation over time.

4.2 'Smart' random walk model

A Smart Random Walk model, SRW, still assumes an underlying random walk (or AR if $w < 1$) process, but also includes provision for stochastic trends based on the high autocorrelation seen previously. The model should be an improvement on the previous model as the continual bias toward crossover should be modeled more correctly. The algorithm is moderately complicated as a weight was given to the observed value, $Ratio_{Obs}$, and the expected value, $Ratio_t$, depending on the significance of the difference between the two.

$$Ratio_{t+1} = w \times Ratio_t + (1 - w) \times Ratio_{Obs} + 0.5 \times (Ratio_{Obs} - Ratio_t) \tag{5}$$

where w represents the weight and $Ratio_t, Ratio_{Obs}$ are the expected and observed ratios respectively.

The weight was obtained by calculating the absolute z-value for the difference between $Ratio_{Obs}$ and $Ratio_t$ and applying the Normal CDF. The normal approximation is a reasonable assumption as the distribution of the proportions at time t is expected to be ∼symmetric, have finite variance, and the sample size is expected to be large enough (well over 30). If an $Ratio_{Obs}$ is closer to $Ratio_t$ then we give the observed ratio less weight than if it was significantly different. The weight, w, is given as follows:

$$w = e^{-abs(z)/2.0}/(2 \times pi \times abs(z)); \tag{6}$$

Where z is calculated by:

$$z = sqrt(n + 1) \times ((Ratio_{Obs} - Ratio_t)/sqrt(Ratio_t \times (1 - Ratio_t))$$

which is simply:

$$z = \frac{(n + 1)(\hat{\theta} - \theta)}{\sqrt{(\theta(1 - \theta))}}$$

If $z > 0.15$ otherwise $w = 1$.
This is to prevent obvious problems with the division by abs(z) causing the weight to increase above 1 - numerically solved to occur at $\sim abs(z) < 0.15$.

The additional move of half the distance between the points forces the algorithm to adjust even if the weight is 1 for the expected value. As the ratio is not expected to be static until after a large number of generations, movement was forced.

4.3 Exponential averaging with global optima

The models produced above assume that the optimal ratio is variable in the short term. However, there is a strong tendency to the upper limit of $95 + \%$ regardless of the starting point. Any short term deviances are not enough to deter from this overall pressure. As a result, can one optimal level be assumed? The assumption is not without merit as one population of source programs should have one optimal value. This model, using Global Optima (GO), should converge more quickly than the random walk models (unless the weight given to the overall average is small), but will not pick up any short term deviances as quickly. In this report, equal weight was given to the overall average as to the previous, expected value (7).

$$Ratio_{t+1} = 0.5 \times Ratio_{t-1} + 0.5 \times OverallAverage_t \tag{7}$$

The best ratio for the next time point is assumed some combination of the observed ratio and the overall average proportion. More information is made use of in this model than in the random walk model as we are combining the results from the differing measurements. In addition, as we are examining a larger population size by grouping the measurements together, the optimal value is likely to be a more accurate estimate.

4.4 Determining improvement over a constant ratio

For the Linear problem, no learning algorithm performed significantly differently from the assumption of a constant operator ratio. This was disappointing but explainable due to the small function set and the relative difficulty, from the GP perspective, of the problem. Therefore, only the results from the non-Linear problem are presented.

The Kolmogorov-Smirnov test can be used to compare the fitness distributions for the two populations as was used earlier. The K-S test is calculated for each generation using the runs as the samples. While the tests between generations are not independent, the two samples compared are independent - satisfying the requirements for the test. Obviously, the results of the test are not expected to be independent in time.

The p-values for the K-S tests: Const 95% versus the RW Algorithm, Const 95% versus the SRW algorithm, and Const 95% versus the GO algorithm, are plotted in Figure 6. Initially the programs in terms of fitness are very similar, but as time increases, the differences between the Learning algorithms and the Const 95% model become more pronounced. The K-S tests shown in Figure 6 clearly show that the

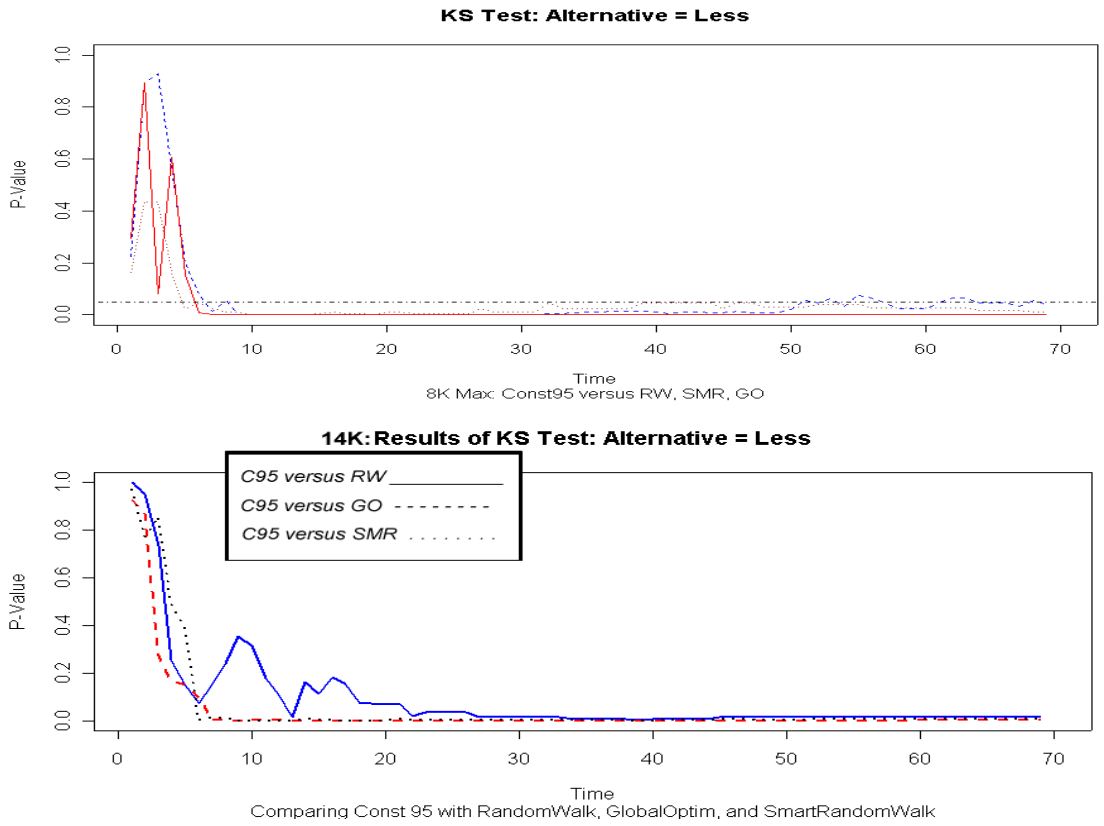

Figure 6: Non-Linear: Performance of Constants versus Variable

fitness has improved - the results from the three learning algorithms are significantly lower (at the 5% level) than the constant model.

It is clear that a variable mutation rate is a significant improvement over a constant rate when the experiment as a whole is examined. A comparison of the learning algorithms to determine the 'best' model yields the results shown in Figure 7.

The KS tests results in Figure 7, show that there is no significant difference between either of the algorithms in the long term - they all end up having minimum values that can not be distinguished by the minimum program fitness distribution. However, between the generations $10 - 35$, the Global Optima method significantly outperforms the Random Walk Algorithm. This is the only difference between the algorithms that can be picked up at a significance level of 5%.

The boxplots in Figure 8 show a similar result for the 8K models. The median as also been added - dashed lines - which show the influence of some of the const 95 runs finishing early. It is evident that similar results would occur for a KS-Test for this run also.

In either algorithm, it is possible for the algorithm not to find an optimal solution - a local optimum is reported instead. However, it is evident from the analyses

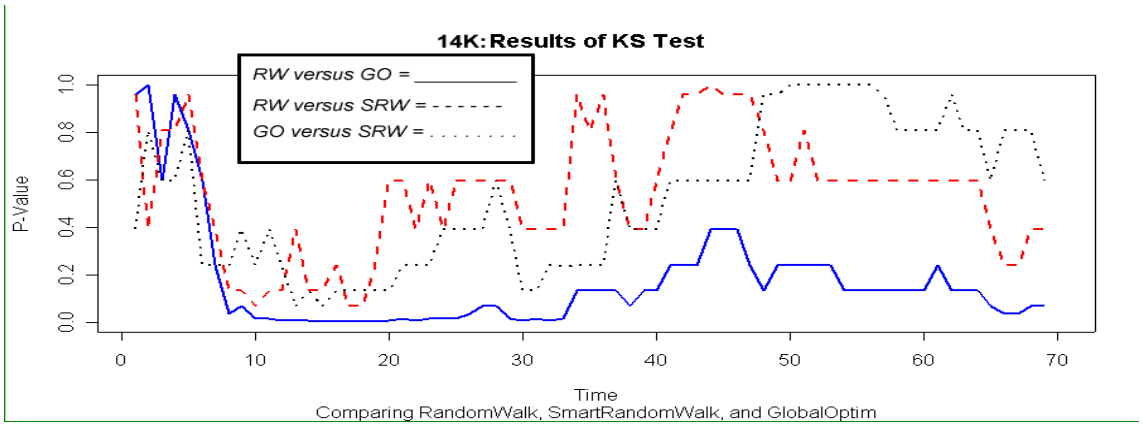

Figure 7: Non-Linear: Comparing Learning KS-Test

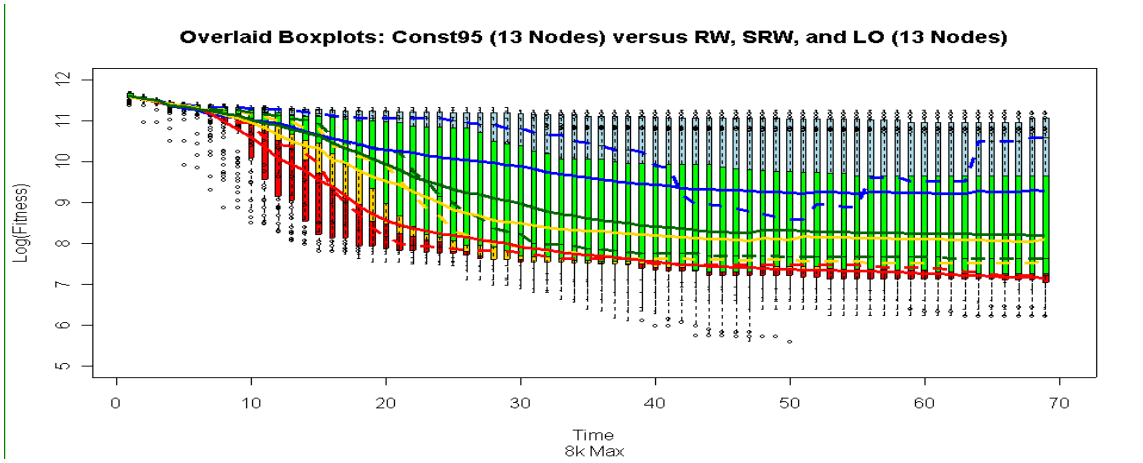

Figure 8: Non-Linear: Comparing Learning KS-Test

performed that the learning algorithms are giving a better result than the constant models as local optima are generally avoided.

5 Conclusions

The operator analysis clearly showed a change in the generated program fitness for the two operators over time. In both cases, non-linear and linear problems, the distribution of the fitness was not constant for either operator. Similarly, the optimal operator levels were not constant but rather tended toward the upper limit of 99% crossover: 1% mutation - especially for the linear problem. Initially, mutation was more helpful than crossover - justifying a higher rate of mutation during the first few generations.

The optimal ratio behaviour was found to vary with time for both problem domains. The general tendency, however, was to tend to the upper limit of 99%

crossover toward 1% mutation. The rate of convergence to this upper limit did vary both with the type of problem and the size of the population on the node.

The assumption of a non-constant ratio did not disadvantage the algorithm when attempting to solve the problem in either domain. It is likely that, for any problem, that any higher rate of mutation initially would be more than compensated by the continual adjustment toward crossover. In the situation when mutation (or crossover!) is very bad, the learning algorithms considered would be able to adjust to this new situation whereas a constant ratio would not.

References

[1] Charles Darwin. *Origin of species by means of natural selection, or the preservation of favoured races in the struggle of life.* London, John Murray, 1872.

[2] Mike J Keith and Martin C Martin. *Advances in Genetic Programming*, chapter 13 Genetic Programming in C++ Implementation Issues. MIT Press, 1994.

[3] John R Koza. *Genetic programming : on the programming of computers by means of natural selection.* Cambridge, Mass. : MIT Press, 1992.

[4] Sean Luke and Lee Spector. A revised comparison of crossover and mutation in genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 208–213, University of Wisconsin, Madison, Wisconsin, USA, 22-25 1998. Morgan Kaufmann.

[5] P D Sniegowski, P J Gerrish, T Johnson, and A Shaver. The evolution of mutation rates: separating causes from consequences. In *BIOESSAYS*, volume 22 (12), pages 1057–1066. John Wiley and Sons, Inc., 2000.